Translating BPMN to Business Rules

Hamda Al-Ali, Ernesto Damiani, Mahmoud Al-Qutayri, Mohammad Abu-Matar, Rabeb Mizouni

Translating BPMN to Business Rules

Hamda Al-Ali[1], Ernesto Damiani[1,2] Mahmoud Al-Qutayri[1], Mohammad Abu-Matar[2], and Rabeb Mizouni[1]

[1] Khalifa University,Abu Dhabi, UAE
{100035242,ernesto.damiani,mqutayri,rabeb.mizouni}@kustar.ac.ae
[2] Etisalat BT Innovation Center (EBTIC), Abu Dhabi, UAE
mohammad.abu-matar@kustar.ac.ae

Abstract. Business Process Model and Notation (BPMN) is a standard graphical notation that is widely used for modeling Business Processes (BP) in Business Process Management (BPM) systems. A key application of such systems is continuous analysis of BP execution for checking compliance of execution logs with process models. In this paper we introduce a simple, human-readable rule language based on a fragment of First-Order Logic (FOL) and show how compliance rules can be generated directly from BPMN models. We focus on control flow aspects of BPMN models by (1) transforming the model to obtain a uniform representation of task activation (2) dividing the model into sets of components and (3) using our proposed language to generate compliance rules for each component. We show that these rules can be used in the analysis of the business process execution log using British Telecom's Aperture business process analysis tool.

Keywords: BPMN, BPM, FOL, Process mining, Conformance checking

1 Introduction

Process mining uses business process logs stored in information systems to gain better understanding about business processes and detect potential breaches between regulations and guidelines specified by process owners and actual execution. In process mining, logs of business event data are compared with the corresponding process model [1]. According to [2] business data are analyzed based on three different perspectives:

- Process perspective: the control flow aspects of the process.
- Organizational perspective: roles of people such as performers and originators of the business processes.
- Case perspective: process-specific properties like location and timing.

The analytics performed on logs depends on the quality and richness of the available data. Activities involved in process mining can be categorized into three types [3]:

- Cartography: includes discovering, enhancement, and diagnosis.

- Auditing: includes detecting, checking, comparing and promoting activities.
- Navigation: includes exploration, prediction and recommendation.

Conformance checking falls under the umbrella of *anomaly detection* because it provides information about mismatches (often called *violations*) between logs and process models and helps process owners to understand the causes. Business rules define constrains or guidelines that apply to an organization. Organizations use business rules to enforce policy, comply with legal obligations, communicate between various parties and perform process analysis. A simple example of a business rule expressed in natural language is *"A customer can pay in cash or by credit card, but a commission of 3$ is applied to credit card payments less than 20$"* . Another example is *"Packages weighing over 20 kg must be shipped by sea"*. In industrial practice, business rules are widely used to assert business structure or to control or influence business behavior [1]. They can apply to people, processes, corporate behavior and computing systems in an organization, and are put in place to help the organization achieve its goals [4].

In this paper, we deal with process-specific rules, i.e. rules that constrain the behavior of a business process in order to achieve a specific goal. These rules can be hidden in source code, inside use cases or in workflow descriptions [5]. Typically, business experts use controlled natural language to write rules based on stake-holders requirements or external regulations, and then compliance experts translate them into the syntax required by the compliance-checking tool used by the organization. Many approaches described in the literature require human intervention for this translation [1]. In this work, we use a simple language to extract logic constraints directly from BPMN models and then translate them into business rules. Using the language, any user can extract the rules easily since they are simple and tool-independent. We build a prototype to extract the logic rules automatically using the XML schema of the BPMN model. Our approach builds on previous work proposed by [6]. In their paper, the authors introduced a mechanism to translate BPMN models to Business Process Execution Language (BPEL), our technique, instead, focuses on the translation of BPMN model to business rules[3]. The rest of the paper is organized as follows. Section 2 highlights conformance checking approaches described in the literature. Section 3 includes various definitions for the BPMN model and its components as well as the logic rules we are proposing. Section 4 presents the case study used to validate our approach. Section 5 concludes the paper and highlights possible future works.

2 Literature Review

Conformance checking is used to detect violations and inconsistencies between a business process' expected behavior and its real-life executions, providing alerts on detected deviations. Conformance checking can use various types of constraints originated from regulations, laws and business process guidelines [6]. Conformance and business concepts may vary according to business domains

[3] Some definitions used in our work are taken verbatim from their paper.

and scenarios. This variety has made it difficult to achieve a fully automated conformance checking approach.

Some research has been done on using logic for conformance checking [7, 8]. Tarantello, Ciccio and Mecella [7] define a declarative process modeling language called *Declare*, which utilizes Linear Temporal Logic (LTL) to define temporal rules to be verified using process logs. The *Declare* language focuses on the control-flow perspective by defining different constraints templates such as $Existence(A)$, $Absence(A)$ and $Response(A, B)$. A pattern-based approach that uses LTL and Metric Temporal Logic (MTL) has been introduced in [9]. The authors define both detective and preventive compliance requirements, which can be used to find violations.

Standard BPMN models can be used to define process behavior as they contain decision logic hardwired in their control flow structure. Experience has shown that logic underlying BPMN models can be employed to express business rules. However, few academic papers have studied the extraction of business rules directly from BPMN models, due to the lack of a standard semantics for BPMN. Most process mining techniques studied in the literature represents processes as Petri nets for analysis and rule generation. In [10] the authors proposed the translation of BPMN models into the Object Management Group (OMG)'s standard Semantics of Business Vocabulary and Rules (SBVR) format, which is used to express business rules within OMG's Model Driven Architecture. The authors map different elements of BPMN model into the SBVR metamodel to extract a SBVR vocabulary. The extracted vocabulary is mapped to controlled English in order to write business rules. In [12], the authors present a method for translating BPMN models into rules in a formalized language called XTT2, which is equipped with a simple partial semantics. Their translation technique is based on visual modeling where similar rules are grouped in a table and a process is divided into a network of tables. However, the XTT2 approach does not target automatic compliance checking. More recently, [13] defined an algorithm for business rules analysis that extracts a *process schema* from the BPMN model and then generates business rules in a syntax called Business Rule Language (BRL). BRL rules are then verified based on the process schema to detect violations. However, the BRL approach only cover a fragment of BPMN as it only deals with "IF" and "THEN" clauses with AND and OR logical operators. Some work is also available on automatic enforcement of business rules in the context of business process. To align business processes with organizations business strategies, the authors in [11] presented SAF, a strategic alignment framework for monitoring organizations. SAF implements monitoring probes, which control the performance of business processes and business strategies achievements. An approach combining BPMN models and logic rules is presented by Awad et al. in [14]. Their main contribution is enabling compliance checking regarding the ordering of activities. A major difference between the approach by Awad et al. and the one proposed in this paper is that they start with a set of externally defined rules and then translate them to temporal logic in order to check the generated model conformance while we start with a BPMN model and use it to

generate business rules. Our choice allows the rule designer to exploit standard BPMN process patterns that are widely available for many industries.

3 Mapping BPMN to FOL

We start by extracting FOL constraints directly from BPMN models. Our constraints will be translated later into business rules. We rely on an initial graph transformation to achieve an implicit uniform task activation semantics; then, we apply the basic definitions given in [6] with some minor variations [4].

3.1 Graph Transformation

We start by translating the BPMN model into a fully synchronous workflow. In BPMN, activity are by default performed synchronously in relation to the invoking process flow, i.e. the process waits for an activity to complete before the process can proceed. However, BPMN syntax allows specifying asynchronous activity execution, e.g. requiring an external event to take place for enabling the execution of an activity. Using asynchronous events (rather than the completion of a previous activity) to enable execution of activities provides a general way to express different enabling semantics. The gist of our transformation is to avoid this complexity by treating synchronization events as special case of ordinary activities, and always use activity enabling by-compilation (of previous activity). In other words, before any analysis, all intermediate events in a BPMN model are transformed to special tasks with double borders to distinguish them. While such transformation may decrease the expressive power of the language, it has the advantage of decreasing the complexity of the model. For the exclusive gateway (XOR), we exclude the default statement, which leads to nothing. The WHILE component will be replaced with REAPEAT to avoid null activity. Summarizing, we perform the following transformations of the BPMN model:

a. Conversion of events into activities.
b. Elimination of DEFAULT in XOR component.
c. Substitution of WHILE with REPEAT component.

As shown in Figure 1, the intermdiate event e_1 is transformed to a task t_2. The DEFAULT sequence flow in the switch component is removed. At the end, the WHILE component is substituted with REPEAT component.

3.2 Business Process Diagram (BPD)

Business processes are expressed graphically using BPMN elements in a BPD. The model is composed of a set of different tasks, events and gateways referred as objects. A task is a single activity in the model while events can represent

[4] The full definitions can be found in [6]

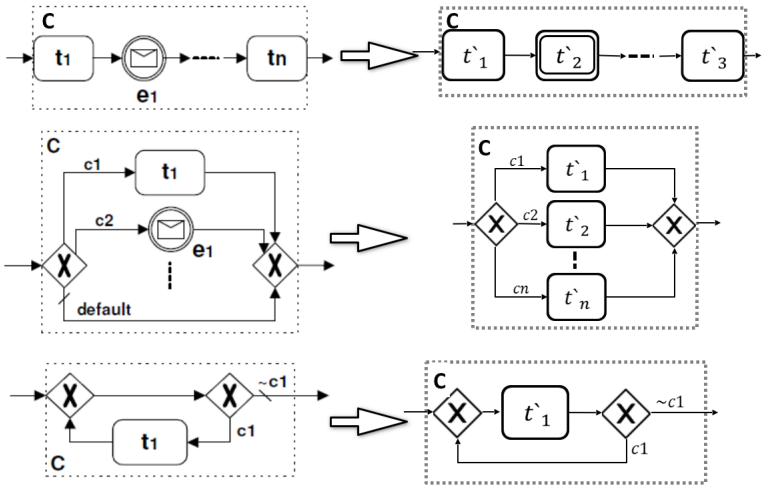

Fig. 1. Intermediate event, elimination of DEFAULT and substitution of WHILE transformation

the start, intermediate, end, and termination of the process (graph transformation will exclude intermediate events), While the gateway represents parallel and XOR forks and joins. Figure 2 shows the graphical representation of some BPMN elements in a core BPD which is composed of set of objects that can be partitioned into disjoint sets of tasks $\mathcal{T}$, events $\mathcal{E}$ and gateways $\mathcal{G}$ [6].

In the remainder of the paper, we only consider well-formed core $\mathcal{BPDs}$ as defined in [6]. Moreover, without losing generality we assume that both $\mathcal{E}^{\mathcal{S}}$ *and* $\mathcal{E}^{\mathcal{E}}$ *are singletons, i.e.* $\mathcal{E}^{\mathcal{S}} = \{s\}$ *and* $\mathcal{E}^{\mathcal{E}} = \{e\}$.

3.3 Decomposing a BPD into components

The notion of component is used to transform a graph structure into set of business rules. To facilitate this transformation, the BPD is decomposed into different components. Again according to [6] "A component is a subset of the BPD that has one entry and one exit point". Each component will be mapped into a single logic rule. Each component should include a minimum number of two different objects (source and sink). A BPD with no component which only contain a single task between the start and end events is called a *trivial* BPD. Whenever we reach a *trivial* BPD, no rule can be extracted and therefore we stop the translation. Breaking down the BPD into set of components helps to define an iterative method to transform BPD into rules. A function **Fold** is defined in [6] which substitutes a component with single task. **Fold** function can be utilized to reduce the BPD iteratively until we reach a *trivial* BPD.

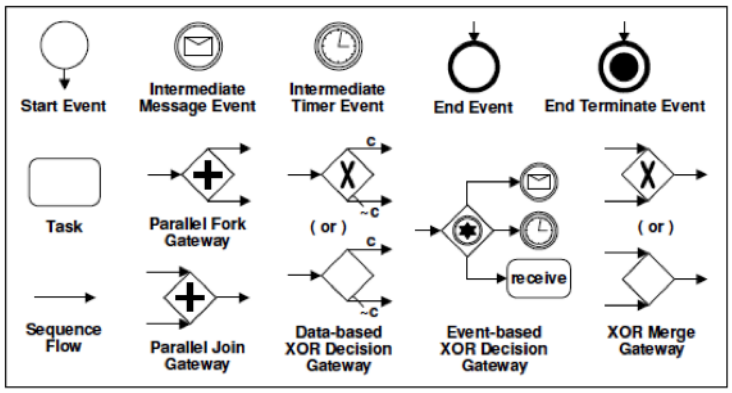

Fig. 2. A core subset of BPMN elements [6]

3.4 Structured activity-based translation

In our approach, different components are mapped into a subset of FOL rules including AND, XOR and sequence operations. Paper [6] defines seven forms of well-structured components. Figure 3 represents the mapping of each component into the corresponding FOL rules [6].

Each rule corresponds to a specific position in the BPD. The position information can be utilized in different ways in the conformance checking process and introduces two different types of dependencies: sequential and hierarchal dependencies. Sequential order means rules extracted from earlier components should be checked before rules from later components. On the other hand, this technique presents the notion of hierarchy of constraints, which to the best of our knowledge, is not well found in the literature. One or more rules can depend on another rule and therefore executing high-level constraints plays critical role in the execution of other low-level constraints.

3.5 Translation Algorithm

After mapping each component to the corresponding rule, we introduce the algorithm used to translation a well-formed core BPD into FOL rule which is similar to the algorithm introduced in [6] with some modifications. The algorithm includes three different steps, selecting a well-structured component then providing its FOL rule and finally fold the component. This is done repeatedly until we reach a *trivial* BPD.

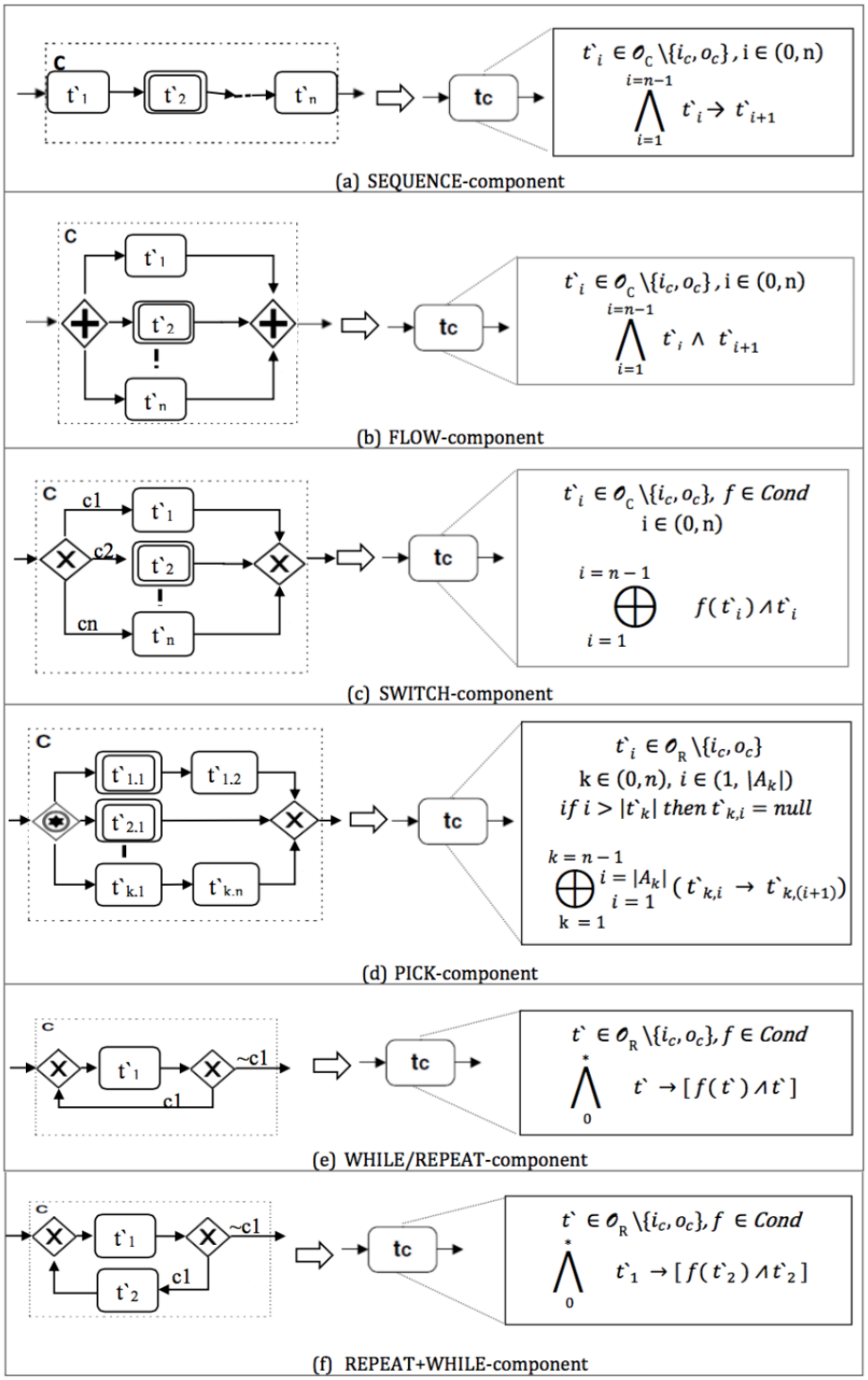

Fig. 3. Folding a well-structured component $\mathcal{C}$ into single task object t_c attached with the corresponding FOL rule translation of $\mathcal{C}$ [6]

Definition 1 (Algorithm 1[6]).

Let $\mathcal{BPD} = (\mathcal{O}, \mathcal{F}, Cond)$ be a well-formed core $\mathcal{BPD}$ with one start event and one end event. $[X]_c$ is the set of components of BPD[X]

1. *$X := \mathcal{BPD}$*
2. *if $[X]_c = \varnothing$ (i.e., X is initially a trivial$\mathcal{BPD}$), stop.*
3. *while $[X]_c \neq \varnothing$ (i.e., X is a non-trivial $\mathcal{BPD}$)*
   - 3.a. *if there is a maximal SEQUENCE component $\mathcal{C} \in [X]_c$, select it and goto (3-c).*
   - 3.b. *if there is a well-structured (non-sequence) component $\mathcal{C} \in [X]_c$, select and goto (3-c).*
   - 3.c. *Attach logic rule translation of $\mathcal{C}$ to task object t_c.*
   - 3.d. *$X :=$ **Fold**$(X, \mathcal{C}, t_c)$ and return to (3).*
4. *Output the logic rule attached to the task object t_c.*

4 Case Study

4.1 BPMN model to logic rules

To validate our approach, we targeted a reference business process in the telecom industrial domain. Processes in this field of industry have been the target of several interesting studies and research [4] because of the challenges they pose to business rules lifecycle management: telecommunication regulation and policies are frequently updated, while actors' roles change over time. We focus on the telecom order fulfillment process in Figure 4 provided by [16] with some modifications. The process starts whenever an *order request is received* and then the process initiator *acknowledge order*. Next, three tasks (*Save order in CM, Send order details to Logistics and Obtain IMSI for ICCID*) are executed in sequence. If the customer chooses to subscribe only then task *Authentication flow* will be fulfilled and if he/she chooses subscription with device then *MNP wait time* is executed. Depending on MNP and new voice/data conditions the sequence flow will branch to *MNP process* or proceed to choose new data or voice. The customer can choose between *Voice subscription process or Data subscription process*. Finally, *Additional service provisioning* is applied followed by *Sending an email notification to customer*.

Based on the algorithm discussed in Section 3 we show how to extract the logic rules from the order fulfillment model. Seven different components are identified as shown in figure 5 and the model is reduced to *trivial BPD*. The components are labeled with number i indicating their order and each C_i is folded into a task t^i_C.

1st Translation The algorithm starts with recognizing the sequence component in the model. The component C_1 with five different tasks is folded into task t^1_C with the following logic rule:

R1: $(ROR \rightarrow AO) \wedge (AO \rightarrow SOC) \wedge (SOC \rightarrow SOD) \wedge (SOD \rightarrow OII)$

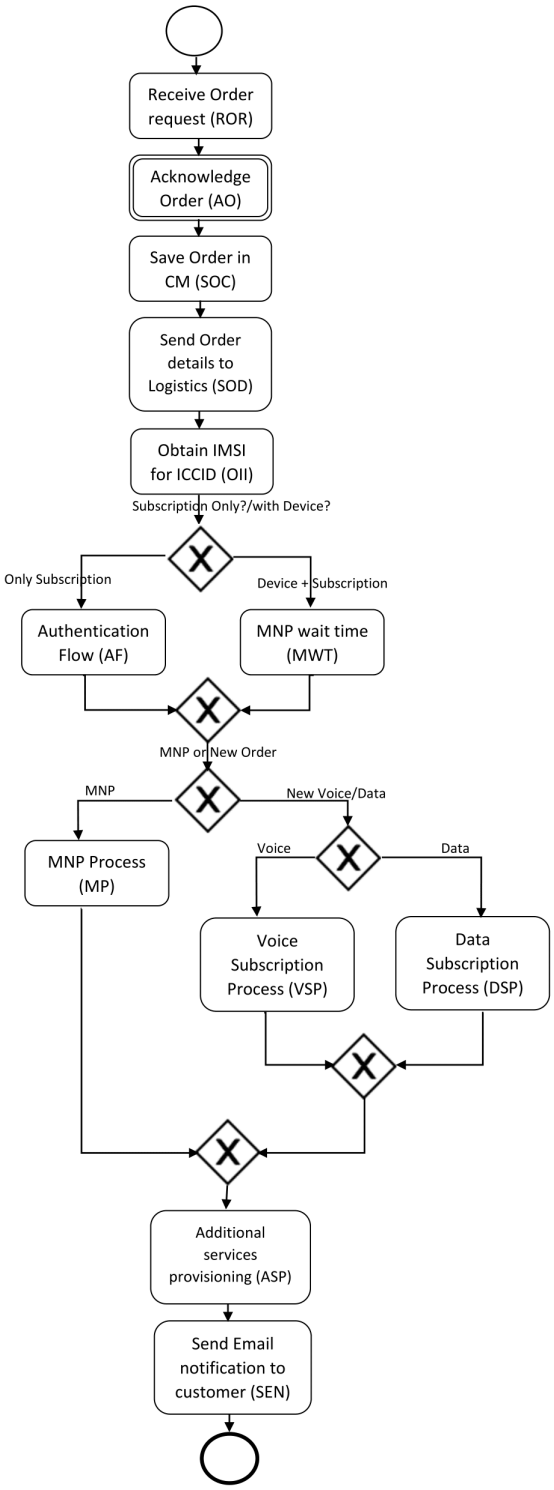

Fig. 4. Order Fulfillment Process [16]

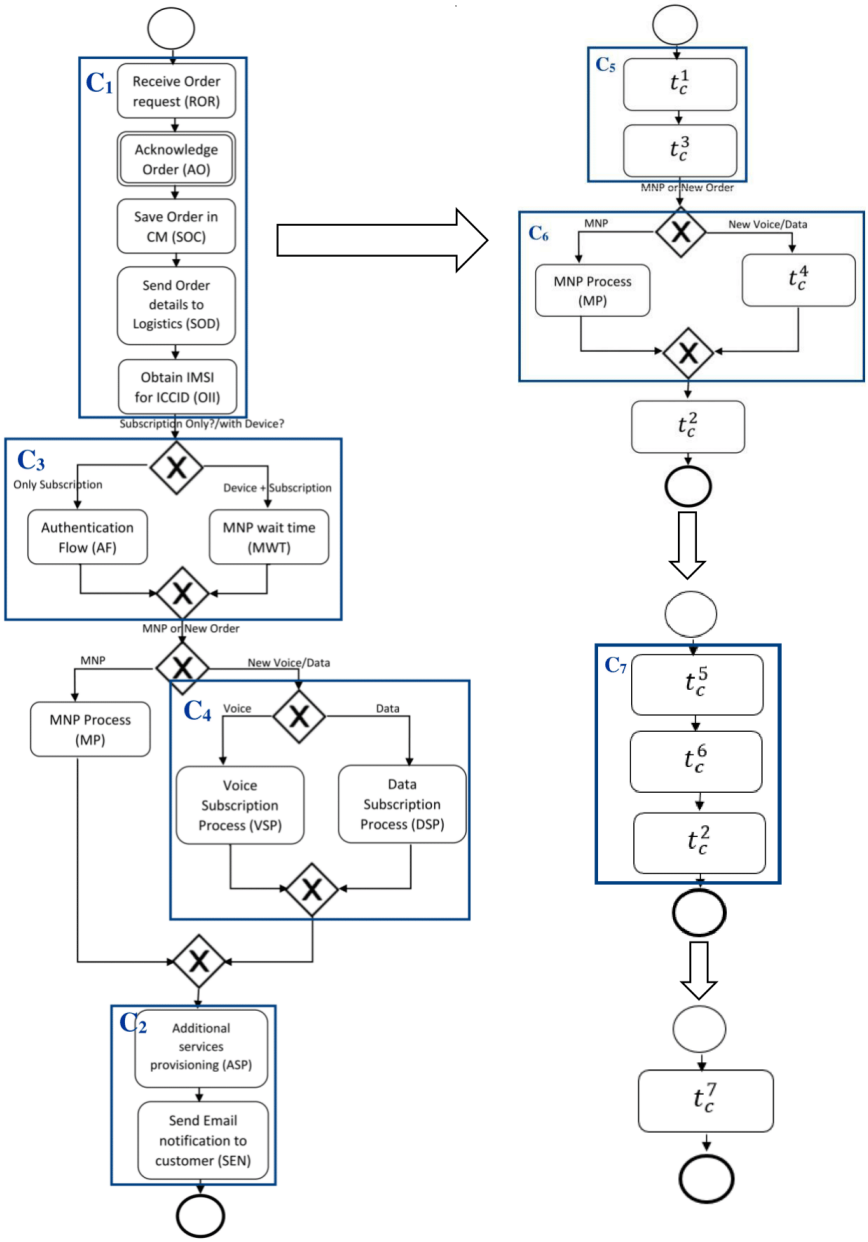

Fig. 5. Translating the order fulfillment process model in Figure 4 into logic rules

2nd Translation After the first fold, another sequence component is identified consisting of tasks *Additional services provisioning* and *send email notification.*The component C_2 is folded into task: t_C^2 attached with the following rule:

$$\textbf{R2: } ASP \rightarrow SEN$$

3rd Translation As no sequence component can be identified in the model, the algorithm tries to allocate any well-structured (non-sequence) component. The switch component C_3 is selected and folded into task t_C^3. The following XOR rule is generated:

$$\textbf{R3: } (Subscription\ only?/with\ Device?Only\ Subscription \wedge AF) \oplus (Subscription\ only?/with\ Device?Device + Subscription \wedge MWT)$$

4th Translation Component C_4 containing the second switch is chosen and folded into task t_C^4 with the corresponding rule:

$$\textbf{R4: } (New\ Voice/Data?Voice \wedge VSP) \oplus (New\ Voice/Data?Data \wedge DSP)$$

5th Translation Translation 1 and 3 introduce new sequence component identified by the algorithm. The new component C_5 is folded into task t_C^5 creating the implication rule:

$$\textbf{R5: } t_C^1 \rightarrow t_C^3$$

6th Translation Folding C_4 simplifies the switch component identified as C_6 that is folded into task t_C^6 and rule 6 is extracted:

$$\textbf{R6: } (MNP\ or\ New\ Order?MNP \wedge MP) \oplus (MNP\ or\ New\ Order?New \wedge t_C^4)$$

7th Translation Finally, the sequence component consisting of t_C^5 , t_C^6 and t_C^2 is folded into task t_C^7 with the following rule:

$$\textbf{R7: } (t_C^5 \rightarrow t_C^6) \wedge (t_C^6 \rightarrow t_C^2)$$

4.2 Evaluation

We implemented our translation algorithm as a Java program which takes as input the XML incarnation of the source BPMN model. The tool parses XML elements to produce a set of rules. We performed some testing to evaluate our translation tool. Namely, we extracted rules from 10 different BPMN models, obtaining a total of 36 business rules. Table 1 shows that the average recall (defined as the percentage of original business process' components that are covered by the generated rules) is 86.1% while the average precision (defined as the percentage of rules that cover a component of the original business process) is 91.1%. We remark that missing rules are associated with folded components therefore the performance of the tool can be increased either by improving the components folding step or by limited human supervision in the translation process.

Table 1. Results of evaluation

Type/Metrics	Nsample	Ncorrect	Nincorrect	Nmissing	Recall	Precision
Business Rules	36	31	3	2	86.1%	91.1%

4.3 Translating FOL into checkable business rules

Aperture [15] is a process mining tool developed by British Telecommunication (BT) group. Its main applications involve the creation of process models from logs of previous process executions, as well as computing Key Performance indicators (KPIs). Using the results of the previous sections, we now add a new feature to Aperture: the capability of checking process conformance. Aperture tool uses its own Workflow Expression language (WEL). WEL is equipped with different operations such as Arithmetic, Relational, Logical and Conditional operations to enable wide range of information extraction. The general format for WEL rules is:

$$[\textit{task-name}].\textit{selection-type@attribute-name}^{5}$$

We need to prove that our language is abstract and independent therefore it can be applied to any tool. For that, we will use WEL to transform the logic rules extracted from the order fulfillment process into WEL business rules that can be directly applied to the tool. Without WHILE or REPEAT control flow structures, all tasks will be executed once and therefore we will use "*first*" as the selection type. Moreover, all tasks have *startTime (sT), endTime (eT) and conditionValue (cV)*. We assume that there is One-to-One correspondence between the tasks in the BPMN model and Aperture's tasks and therefore we do not need any log lifting. Below, we define our algorithm to translate FOL rules extracted from the BPMN model to Aperture's WEL rules. We start by selecting a rule, finding its equivalent WEL rule and output it.

Definition 2 (Algorithm 2).
Let $\mathcal{BPD} = (\mathcal{O}, \mathcal{F}, Cond)$ be a well-formed core $\mathcal{BPD}$. $[X]_R$ is the set of logic rules of BPD[X].

1. $X := \mathcal{BPD}$
2. *if* $[X]_R = \varnothing$, *stop.*
3. *while* $[X]_R \neq \varnothing$
   R-WEL: *(Aperture rule corresponding to logic rule R)*
   3.a. ***if*** $R \mathrel{\widehat{=}}$ *SEQUENCE-component* ***then***
   for $i{=}1 \rightarrow |t|$ ***do***
   $$[t_i].type@eT < [t_{i+1}].type@sT$$
   3.b. ***if*** $R \mathrel{\widehat{=}}$ *FLOW-component* ***then***
   for $i{=}1 \rightarrow |t|$ ***do***

[5] The selection type refers to the time the task was executed in the process, it can be first, last or any

$[t_i].type@sT <= [t_{i+1}].type@eT$ & $[t_{i+1}].type@sT <= [t_i].type@eT$

3.c. ***if*** $R \mathrel{\widehat{=}}$ *SWITCH-component* ***then***
for $i=1 \rightarrow |t|$ ***do***
(($[PA].type@cV = f(t_i)$ & $[t_i].type@sT > 0$) | ($[PA].type@cV = f(t_{i+1})$ & $[t_{i+1}].type@sT > 0$)) &! (($[PA].type@cV = f(t_i)$ &$[t_i].type@sT > 0$) & ($[PA].type@cV = f(t_{i+1})$ & $[t_{i+1}].type@sT > 0$))

3.d. ***if*** $R \mathrel{\widehat{=}}$ *PICK-component* ***then***
for $k=1 \rightarrow n$ ***do***
for $i=1 \rightarrow |t|$ ***do***
(($[t_{k,i}].type@eT < [t_{k,(i+1)}].type@sT$) | ($[t_{k+1,i}].type@eT < [t_{k+1,(i+1)}].type@sT$)) &! (($[t_{k,i}].type@eT < [t_{k,(i+1)}].type@sT$) & ($[t_{k+1,i}].type@eT < [t_{k+1,(i+1)}].type@sT$))

3.e. ***if*** $R \mathrel{\widehat{=}}$ *WHILE/REPEAT-component* ***then***
for $i=0 \rightarrow *$ ***do***
$[t_1].type@eT < f(t_1)$ & $[t_1].type@sT$

3.f. ***if*** $R \mathrel{\widehat{=}}$ *REPEAT+WHILE-component* ***then***
for $i=0 \rightarrow *$ ***do***
$[t_1].type@eT < ([t_1].type@cV = f(t_2)$ & $[t_2].type@sT$

4. *Output Aperture WEL rule.*

Using the above algorithm, we transform the rules found in subsection 4.1 into Aperture accepted format.

R1-WEL: ([ROR].first@eT < [AO].first@sT) & ([AO].first@eT < [SOC].first@sT) & ([SOC].first@eT < [SOD].first@sT) & ([SOD].first@eT < [OII].first@sT)

R2-WEL: ([ASP].first@eT < [SEN].first@sT)

R3-WEL: ([OII].first@cV == 'Only Subscription' & [AF].first@sT > 0) | ([OII].first@cV == 'Device + Subscription' & [MWT].first@sT > 0) & ! ([OII].first@cV == 'Only Subscription' & [AF].first@sT > 0) & ([OII].first@cV == 'Device + Subscription' & [MWT].first@sT > 0)

R4-WEL: ($[t^3_C]$.first@cV == 'New Voice' & [VSP].first@sT> 0) | ($[t^3_C]$.first@cV == 'New Data' & [DSP].first@sT> 0) & ! ($[t^3_C]$.first@cV == 'New Voice' & [VSP].first@sT> 0) & ! ($[t^3_C]$.first@cV == 'New Data' & [VSP].first@sT> 0)

R5-WEL: ($[t^1_C]$.first@eT < $[t^3_C]$.first@sT)

R6-WEL: ($[t^3_C]$.first@cV == 'MNP' & [MP].first@sT> 0) | ($[t^3_C]$.first@cV == 'New order' &$[t^4_C]$.first@sT> 0) & ! ($[t^3_C]$.first@cV == 'MNP' & [MP].first@sT> 0) & ($[t^3_C]$.first@cV == 'New order' & $[t^4_C]$.first@sT> 0)

R7-WEL: ($[t^5_C]$.first@eT < $[t^6_C]$.first@sT) & ($[t^6_C]$.first@eT < $[t^2_C]$.first@sT)

The translated business rules are independent from the log and therefore we kept tasks t^i_C. These tasks represent temporary tasks and using the log later we will be able to replace them with the corresponding task. For example, t^3_C in R4-WEL will be replaced with either task AF or task MWT based on which tasks was executed and founded in the log.

5 Conclusion

In this paper, we introduced a pragmatic translation of BPMN models into simple FOL rules reflecting the control flow aspects of the model. Our strategy is based on uniform representation of asynchronous task activation via special synchronous tasks that correspond to checking event queues. The rules we obtain are independent from any software tool and can be used as a baseline to write conformance rules without the need of business expert intervention. In order to validate our approach we extracted business rules from sample BPMN models. These rules were transformed later into the syntax of Aperture, BT's industrial process mining tool. We described and tested an implementation of our translation algorithm.

Future work will involve full automation of the translation process. Components' information can be further utilized to explore the sequential and hierarchal dependencies. This will help ensure that conformance rules are applied in the right time and order.

References

1. Van der Aalst, W., Damiani, E.: Processes Meet Big Data: Connecting Data Science with Process Science. IEEE Transactions on Services Computing. 8, 810-819 (2015).
2. Van der Aalst, W., Reijers, H., Weijters, A., Van Dongen, B., Alves de Medeiros, A., Song, M., Verbeek, H.: Business process mining: An industrial application. Information Systems. 32, 713-732 (2007)
3. Ly, L., Maggi, F., Montali, M., Rinderle-Ma, S., Van der Aalst, W.: Compliance monitoring in business processes: Functionalities, application, and tool-support. Information Systems. 54, 209-234 (2015).
4. Huurros, M.: The emergence and scope of complex system / service innovation: the case of the mobile payment service market in Finland. Helsinki School of Economics (2007).
5. Von Halle, B., Goldberg, L., Zachman, J.: The business rule revolution. HappyAbout.info, Cupertino, 9-10 CA (2006).
6. Ouyang, C., Van der Aalst, W., Dumas, M., Ter Hofstede, A.H.M.: Translating BPMN to BPEL. Technical Report BPM-06-02, BPM Center (2006).
7. Tarantello, G., Ciccio, C., Mecella, M.: On the Discovery of Declarative Control Flows for Artful Processes. ACM Transactions on Management Information Systems. 5, 1-37 (2015).
8. Burattin, A., Cimitile, M., Maggi, F., Sperduti, A.: Online Discovery of Declarative Process Models from Event Streams. IEEE Transactions on Services Computing. 8, 833-846 (2015).

9. Turetken, O., Elgammal, A., Van den Heuvel, W., Papazoglou, M.: Capturing Compliance Requirements: A Pattern-Based Approach. IEEE Software. 29, 28-36 (2012).
10. Malik, S., Bajwa, I.: A Rule Based Approach for Business Rule Generation from Business Process Models. Rules on the Web: Research and Applications. 92-99 (2012).
11. Damiani E., Mulazzani F., Russo B., Succi G. (2008) SAF: Strategic Alignment Framework for Monitoring Organizations. In: Abramowicz W., Fensel D. (eds) Business Information Systems. BIS 2008. Lecture Notes in Business Information Processing, vol 7. Springer, Berlin, Heidelberg
12. Kluza, K., Malanka, T., Nalepa, G., Ligza, A.: Proposal of Representing BPMN Diagrams with XTT2-Based Business Rules. Intelligent Distributed Computing V. 243-248 (2011).
13. Rachdi, A., En-Nouaary, A., Dahchour, M.: Analysis of common business rules in BPMN process models using business rule language. 2016 11th International Conference on Intelligent Systems: Theories and Applications (SITA). (2016).
14. Awad, A., Decker, G., Weske, M.: Efficient Compliance Checking Using BPMN-Q and Temporal Logic. Lecture Notes in Computer Science. 326-341
15. Taylor, P., Leida, M., Majeed, B.: Case Study in Process Mining in a Multinational Enterprise. Lecture Notes in Business Information Processing. 134-153 (2012).
16. Rowley, M., Minnam, S., Koppala, V.: Using BPM to Implement Services in Telecom and Media: Practical Advice and Insights from the Real World. ActiveVOS 30-33 (2012).